Homework 1

Problem 1: One-step error probability

For the first part of the problem I wrote the following programm in MATLAB:

```
p = [12, 24, 48, 70, 100, 120];
N = 120;
perror1 = [];
perror2 = [];
for m = 1:2
  perror = [];
  for i = 1:length(p)
    e=0;
    for j = 1:10^5
      pattern = 2*randi([0,1],N,p(i)) - 1;
      W = 0;
      for k = 1:size(pattern, 2)
        w = w + mtimes(pattern(:,k),transpose(pattern(:,k)));
      end
      if m == 1
        W = W - diag(diag(W));
      end
      W = W/N;
      randPattern = pattern(:,randi(p(i)));
      randBit = randi(N);
      b = mtimes(w,randPattern);
      x_{temp} = sign(b(randBit));
      if x_temp ~= randPattern(randBit)
        e = e + 1;
      end
    end % trials loop
    perror = [perror e/10^5];
  end % patterns loop
  if m == 1
    perror1 = perror;
    perror2 = perror;
  end
end
```

Problem 2: Recognising digits

For the second problem I wrote the following programm in MATLAB:

```
fivepatterns % runs the fivepatterns.m file to import its values
patterns=[x1; x2; x3; x4; x5;];
N = size(patterns, 2);
patterns = transpose(patterns);
W=0;
for i = 1:size(patterns, 2)
    w = w + mtimes(patterns(:,i),transpose(patterns(:,i)));
W = W - diag(diag(W));
W = W/N;
1, 1, 1, -1], [1, 1, 1, 1, -1, 1, 1, 1, -1], [1, 1, 1, 1, 1, 1, 1, 1, 1, -1], [1, 1, 1, 1, -1, 1, 1, 1, 1, -1], [1, 1, 1, 1, 1, 1, 1, 1, -1], [1, 1, 1, 1, 1, 1, 1, 1, 1, -1], [1, -1, 1, 1, 1, -1, -1], [1, -1, 1, 1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1, -1], [1, -1, -1], [1, -1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1, -1], [1, -1], [1, -1], [1, -1], [1, -1], [1, -1], [1, -1], [1, -1], [1
1, -1, 1, -1, -1, -1, -1]],
                                       [[1, 1, 1, 1, 1, 1, 1, 1, 1], [-1, -1, -1, 1, 1, 1, 1, -1,
 -1, -1], [-1, -1, 1, 1, 1, 1, 1, -1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1]
1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1,
1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1,
-1], [-1, -1, 1, 1, 1, 1, 1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1]
testPatternsOutput = zeros(3, N);
flag = 0;
for l = 1:3
    x_test = testPatterns(l,:);
    T = 100;
    for j = 1:T
         if flag == 1
             flag = 0;
             break
         end
         for k = 1:N
             if flag == 1
                 break
             end
             b = w*transpose(x_test);
             x_{test(k)} = sign(b(k));
         for i=1:size(patterns, 2)
             if transpose(x_test)==patterns(:,i)
                  fprintf('It converges to pattern %d.\n', i);
                  testPatternsOutput(l,:)=transpose(patterns(:,i));
                  flag=1;
```

```
elseif transpose(x_test)==flip(patterns(:,i))
                        fprintf('It converges to pattern %d.\n', -i);
                        testPatternsOutput(l,:)=transpose(patterns(:,i));
                        flag=1;
                        break
                  else
                        flag = 2;
                  end
            end
      end
       if flag == 2
            fprintf('It converges to none of the patterns\n');
             testPatternsOutput(l,:)=x_test;
       end
 end
 where the "fivepatterns.m" file contains the following data:
X1=[ [ -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], [ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1], [ -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1], [ -1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1, -1, 1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [ -1, 1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1], [ -1, 1, 1, 1, -1, -1], [ -1, 1, 1, 1, -1, -1], [ -1, 1, 1, 1, -1, -1], [ -1, -1, 1, 1, 1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -1, -1, -1], [ -
 1, -1, -1, -1, -1, -1, -1, -1] ];
 x2=[ [ -1, -1, -1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -
 1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[
x3=[ [ 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -
 1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1,
 -1, -1, -1, 1, 1, 1, -1, -1], [-1, -1, -1, -1, 1, 1, 1, -1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 
 1, -1, -1, 1, 1, 1, -1, -1],[1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[1, 1, 1, 1, 1,
 1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1,
 -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, -1,
 -1],[ 1, 1, 1, 1, 1, 1, 1, -1, -1] ];
 X4=[ [ -1, -1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, 1, -1],[ -
 1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1,
 -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, 1, -1],[ -1, -
 1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1,
 1, 1, 1, 1, 1, -1, -1], [ -1, -1, -1, -1, -1, 1, 1, 1, -1], [ -1, -1, -1, -1,
 -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1
 1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1,
 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1] ];
 x5=[ [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1], [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1]
 1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[
 -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1,
 1, 1, -1, -1, -1, -1, 1, 1, -1], [ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1], [ -1, 1, 1, 1,
 1, 1, 1, 1, 1, -1],[-1, -1, -1, -1, -1, -1, 1, 1, -1],[-1, -1, -1, -1, -1,
 -1, -1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, -1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
```

Problem 3: Stochastic Hopfield network

For the third problem I wrote the following programm in MATLAB:

```
p = [7, 45];
beta = 2;
N = 200;
Mav = zeros(1, size(p, 2));
for l = 1:size(p,2)
  Msum = 0;
 M=0;
for j = 1:100
    patterns = [];
    for i = 1:p(l)
      patterns = 2*randi([0,1],N,i)-1;
    end
    w = 0;
    for k = 1:p(l)
      w = w + mtimes(patterns(:,k), transpose(patterns(:,k)));
    end
    W = W-diag(diag(W));
    w = w/N;
    x_temp = patterns(:,1);
    S = 0;
    for m = 1:10^3
      for n = 1:N
        b = mtimes(w,x_temp);
        r = rand;
        if r<1/(1+exp(1)^{(-2*beta*b(n))})
          x_{temp(n)} = 1;
        else
          x_{temp(n)} = -1;
      end % bits loop
      S = S + x_{temp};
    end % time loop
    M = dot(S, patterns(:,1));
    Msum = Msum + M/(N*10^3);
  end % trials loop
  Mav(l) = Msum/100;
end
```